Tarea Programada #02

Estudiantes:

- **Emmanuel D. Solis**
- **Josef Ruzicka**
- **Paula Monge**

Descripción del problema.

Bin Packing es un problema de optimización en el que los artículos de diferentes tamaños deben empaquetarse en un número finito de cajas o contenedores, `este tipo de problemas puede tener diversos casos, el que se va a usar para el desarrollo del trabajo es *"Identical-machines scheduling"*.

Identical-machines scheduling es un caso especial de programación de máquinas uniforme, que en sí misma es un caso especial de programación óptima de trabajos, en el cual el tiempo de procesamiento de cada trabajo es el mismo en cada máquina. Se tiene por entrada "n" trabajos de tiempos de procesamiento variables, que deben programarse en "m" máquinas idénticas, de modo que se optimice una determinada función objetivo (la función puede ser minimizar o maximizar). Tomando en cuenta la explicación anterior, la idea del programa sería un "Repartidor de tareas". el cual se encargará de tomar la entrada de "n" tareas y distribuirlo en "m" procesadores, donde la función objetivo es minimizar el tiempo que se tardará en terminar la ejecución de las tareas totales.

Heurística.

Cada computadora simulada contará con una capacidad máxima de pesos que puede soportar, y el peso de cada tarea corresponderá a la cantidad de tiempo que requiera para ser completada.

En la simulación existirá un repartidor de las distintas tareas entre las computadoras, la heurística en la que dicho repartidor se basará consiste en tomar cada tarea e intentar colocarla en la cola menos llena en peso, recorriendo cada una de las colas de tareas de las distintas computadoras.

El fin de esta heurística es distribuir tan equitativamente como sea posible la carga de tareas entre las distintas computadoras con el fin de que todas las tareas sean completadas tan rápido como sea posible. El problema de la heurística es que por cada tarea que se desea distribuir se deben revisar todas las colas de las diferentes computadoras y esto es un proceso muy lento con suficientes computadoras y tareas.

Metaheurística.

La metaheurística escogida es el uso de un algoritmo genético, la idea es que basado en la naturaleza del problema, incluyendo la limitación de cada procesador (sea minimizar o maximizar), se harán cruces con todas las tareas, esto en el supuesto de que ya conocemos todas las tareas que hay que procesar previo a iniciar el algoritmo.

Primero haremos una lista de todas las tareas a procesar, no necesitan estar ordenados, luego iremos escogiendo de uno en uno para hacer la generación inicial de forma pseudoaleatoria, esto porque se escoge de forma aleatoria una tarea basándose en escoger un índice al azar, este irá en primer procesador que se esté llenando, pero si con agregar este número se sobrepasa la capacidad del procesador entonces se descarta y se escoge otro número, si después de recorrer todas las tareas no hay otro que alcance entonces se considerará que este procesador está lleno. Esto respecto a la creación de las primeras soluciones aleatorias, para etapas posteriores se tomarán ya dichas soluciones y se mezclarán; tomando en cuenta hacer siempre una mutación para evitar mejores locales, y escogiendo posiciones para mezclar al azar siempre verificando que no se sobrepasa la capacidad de procesamiento.

Se medirá qué tan buena es una solución en base a la cantidad de procesamiento desperdiciado (en caso de ser minimización) o por cantidad de procesadores usados (en caso de ser maximización), esto se comparará con la solución de la etapa anterior, si es mejor la nueva entonces se reemplazará, de lo contrario se considera que se ha llegado al fin del algoritmo y se entrega como resultado la solución anterior.

La idea de implementar este algoritmo genético es que se pueda distribuir de la forma más equitativa posible toda la carga del problema, sin tener que hacerlo por fuerza bruta probando toda posible combinación, se ha demostrado que los algoritmos genéticos bien implementados ofrecen soluciones muy buenas, razón por la cuál lo escogimos.

Resultados.

Se comparan los 3 algoritmos para así evidenciar cuál de ellos tiene un mejor rendimiento, tanto en tiempo como en distribución de tareas. A continuación se muestran tablas con el rendimiento de cada uno de ellos en una cantidad de 10 ejecuciones para cada algoritmo:

Fuerza Bruta:

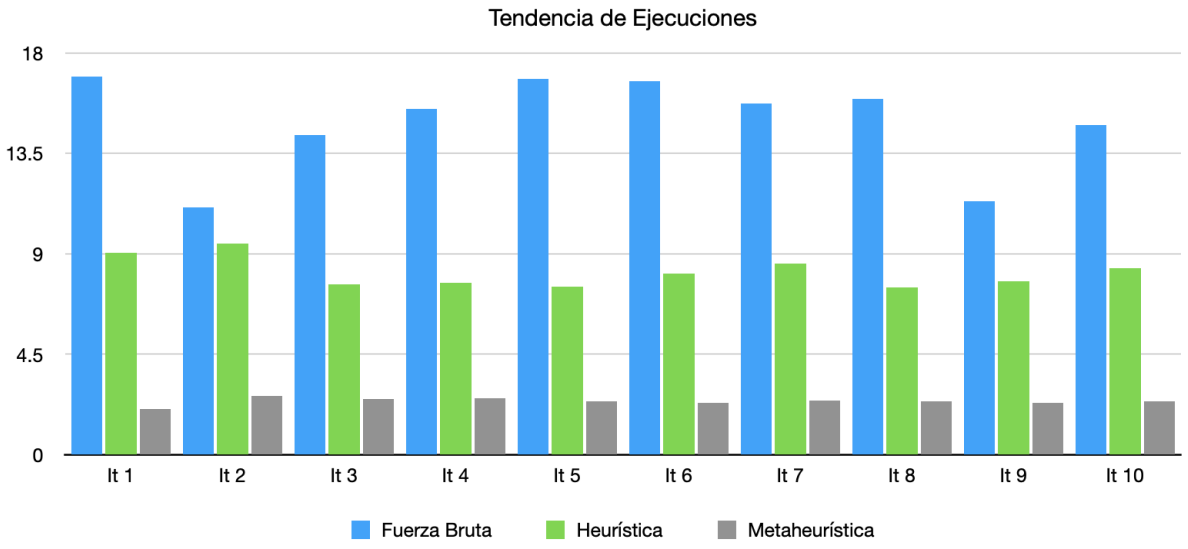
It.1	It.2	It.3	It.4	It.5	It.6	It.7	It.8	It.9	It.10
16.92	11.06	14.33	15.48	16.83	16.73	15.74	15.92	11.34	14.75

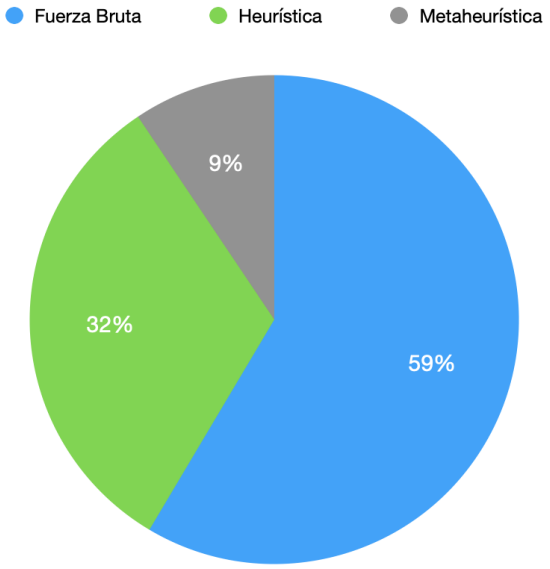
Heurística

It.1	It.2	It.3	It.4	It.5	It.6	It.7	It.8	It.9	It.10
9.06	9.44	7.64	7.71	7.53	8.12	8.57	7.48	7.78	8.35

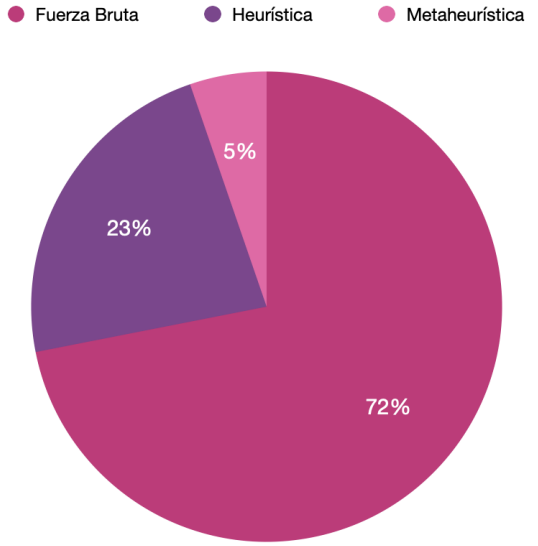
Metaheurística

It.1	It.2	It.3	It.4	It.5	It.6	It.7	It.8	It.9	It.10
2.04	2.63	2.51	2.52	2.40	2.32	2.42	2.40	2.33	2.40





Promedio



Desviación Estándar